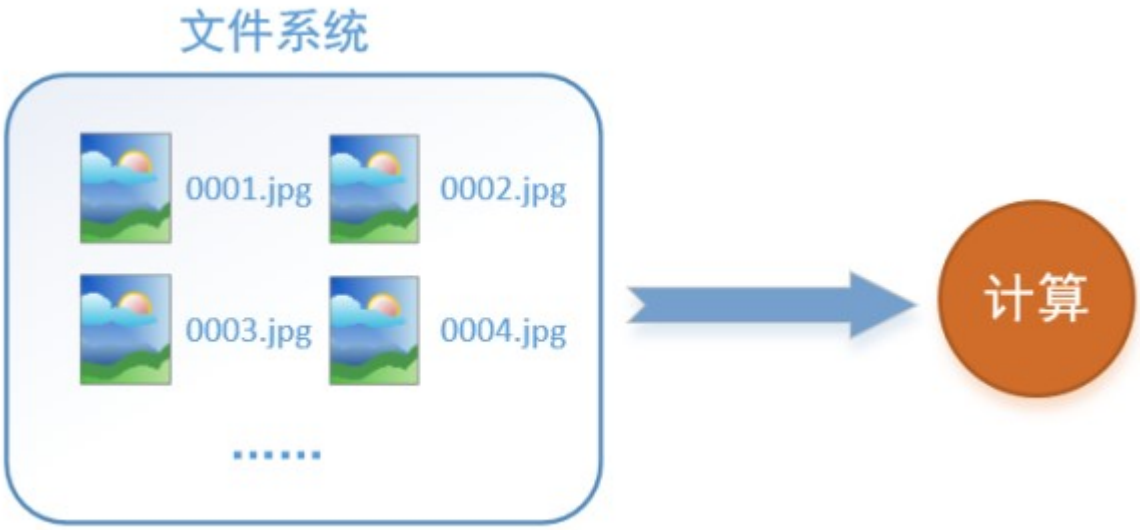
tensorflow 1.0 学习：十图详解tensorflow数据读取机制

本文转自：https://zhuanlan.zhihu.com/p/27238630

在学习tensorflow的过程中，有很多小伙伴反映读取数据这一块很难理解。确实这一块官方的教程比较简略，网上也找不到什么合适的学习材料。今天这篇文章就以图片的形式，用最简单的语言，为大家详细解释一下tensorflow的数据读取机制，文章的最后还会给出实战代码以供参考。

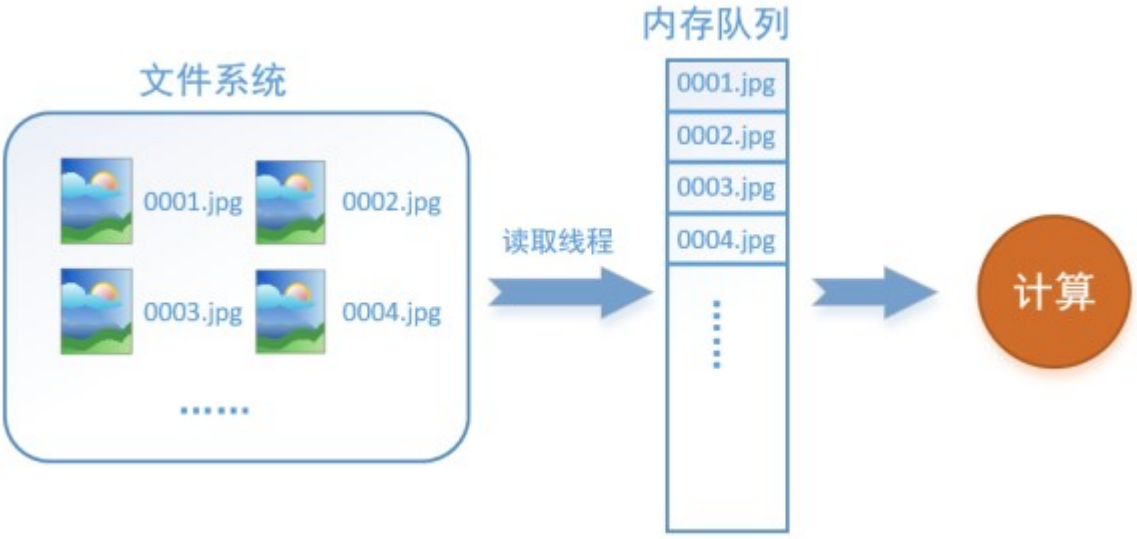
一、tensorflow读取机制图解

首先需要思考的一个问题是，什么是数据读取？以图像数据为例，读取数据的过程可以用下图来表示：



假设我们的硬盘中有一个图片数据集0001.jpg，0002.jpg，0003.jpg……我们只需要把它们读取到内存中，然后提供给GPU或是CPU进行计算就可以了。这听起来很容易，但事实远没有那么简单。**事实上，我们必须要把数据先读入后才能进行计算，假设读入用时0.1s，计算用时0.9s，那么就意味着每过1s，GPU都会有0.1s无事可做，这就大大降低了运算的效率。**

如何解决这个问题？方法就是将读入数据和计算分别放在两个线程中，将数据读入内存的一个队列，如下图所示：

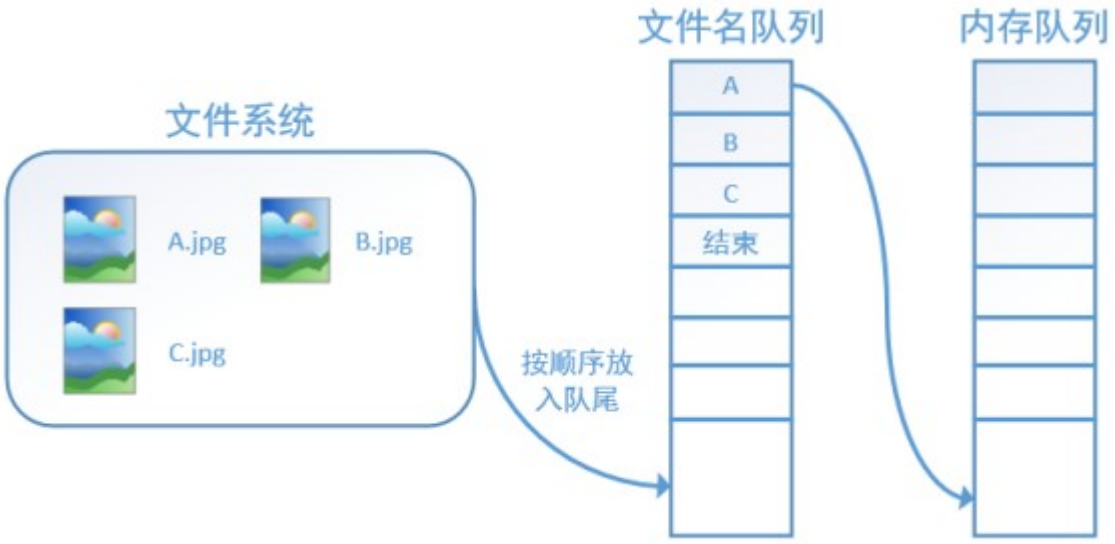


读取线程源源不断地将文件系统中的图片读入到一个内存的队列中，而负责计算的是另一个线程，计算需要数据时，直接从内存队列中取就可以了。这样就可以解决GPU因为IO而空闲的问题！

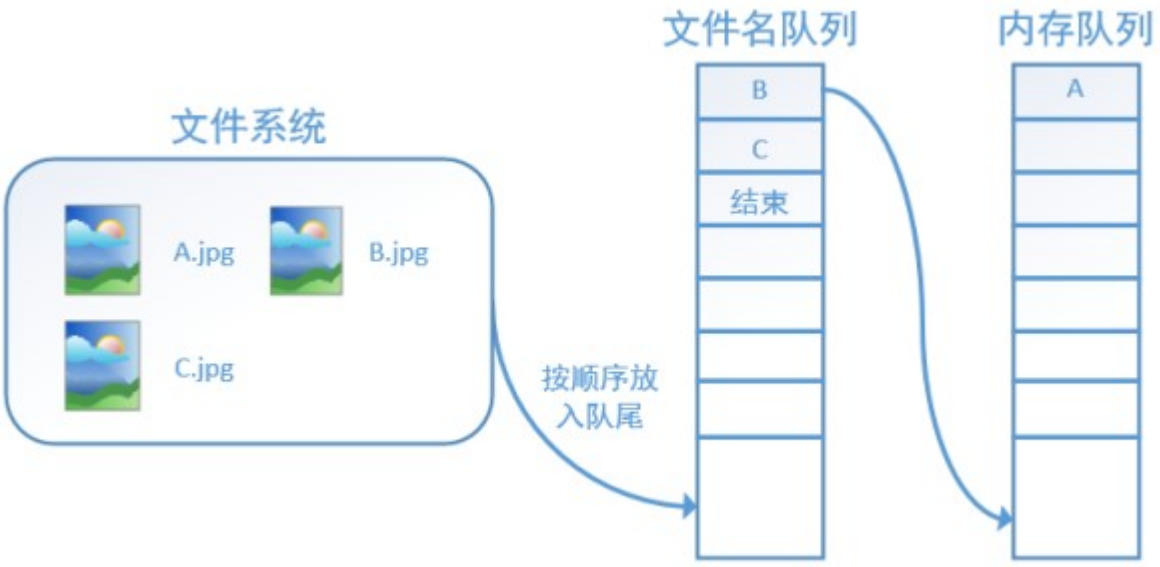
而在tensorflow中，为了方便管理，在内存队列前又添加了一层所谓的**“文件名队列”**。

为什么要添加这一层文件名队列？我们首先得了解机器学习中的一个概念：epoch。对于一个数据集来讲，运行一个epoch就是将这个数据集中的图片全部计算一遍。如一个数据集中有三张图片A.jpg、B.jpg、C.jpg，那么跑一个epoch就是指对A、B、C三张图片都计算了一遍。两个epoch就是指先对A、B、C各计算一遍，然后再全部计算一遍，也就是说每张图片都计算了两遍。

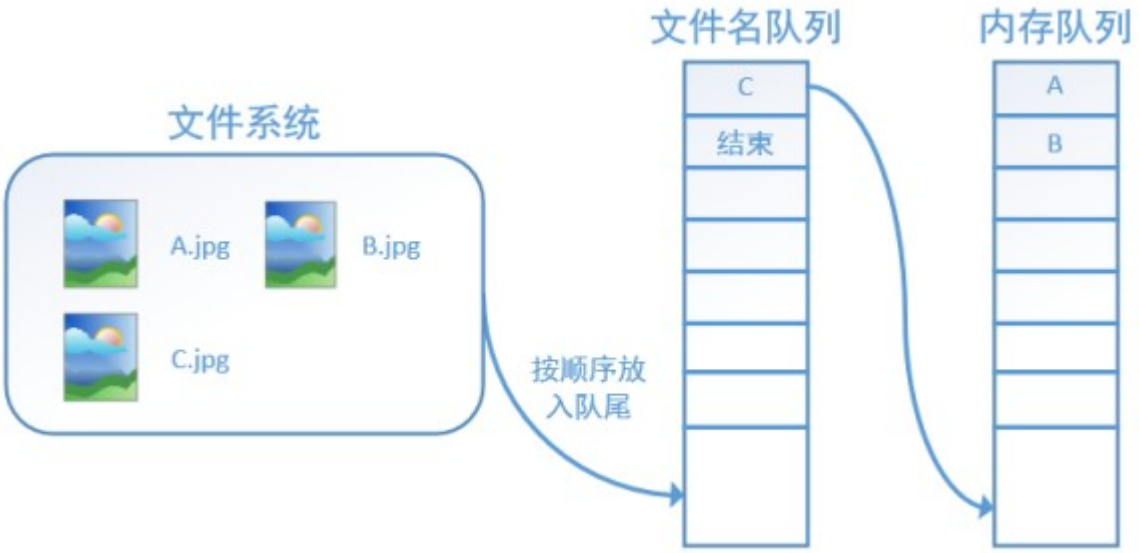
tensorflow使用文件名队列+内存队列双队列的形式读入文件，可以很好地管理epoch。下面我们用图片的形式来说明这个机制的运行方式。如下图，还是以数据集A.jpg, B.jpg, C.jpg为例，假定我们要跑一个epoch，那么我们就在文件名队列中把A、B、C各放入一次，并在之后标注队列结束。

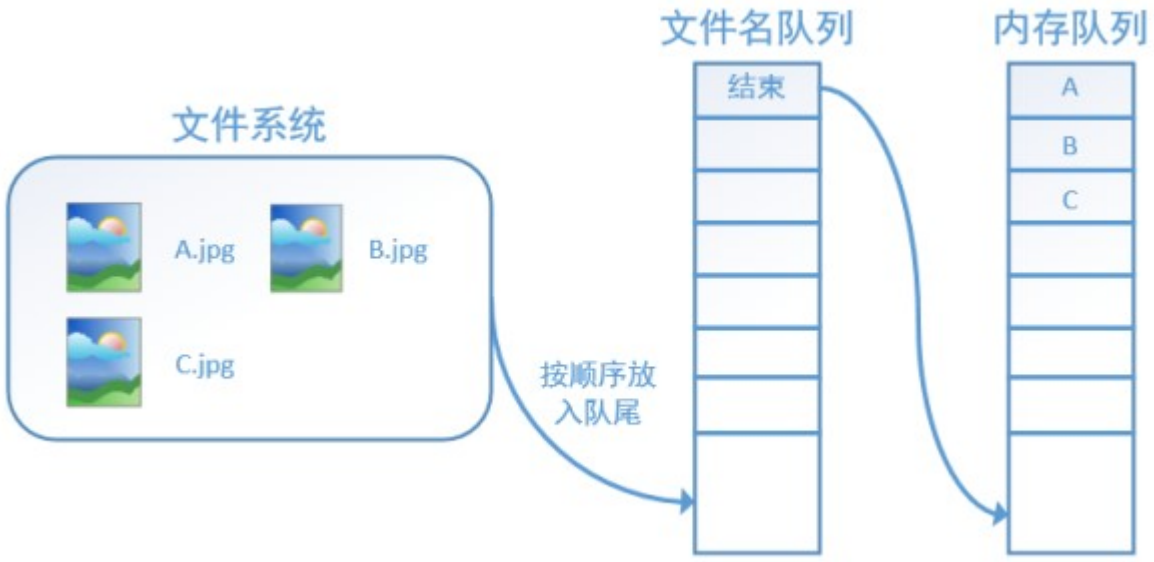


程序运行后，内存队列首先读入A（此时A从文件名队列中出队）：



再依次读入B和C：





此时，如果再尝试读入，系统由于检测到了"结束"，就会自动抛出一个异常（OutOfRange）。外部捕捉到这个异常后就可以结束程序了。这就是tensorflow中读取数据的基本机制。如果我们要跑2个epoch而不是1个epoch，那

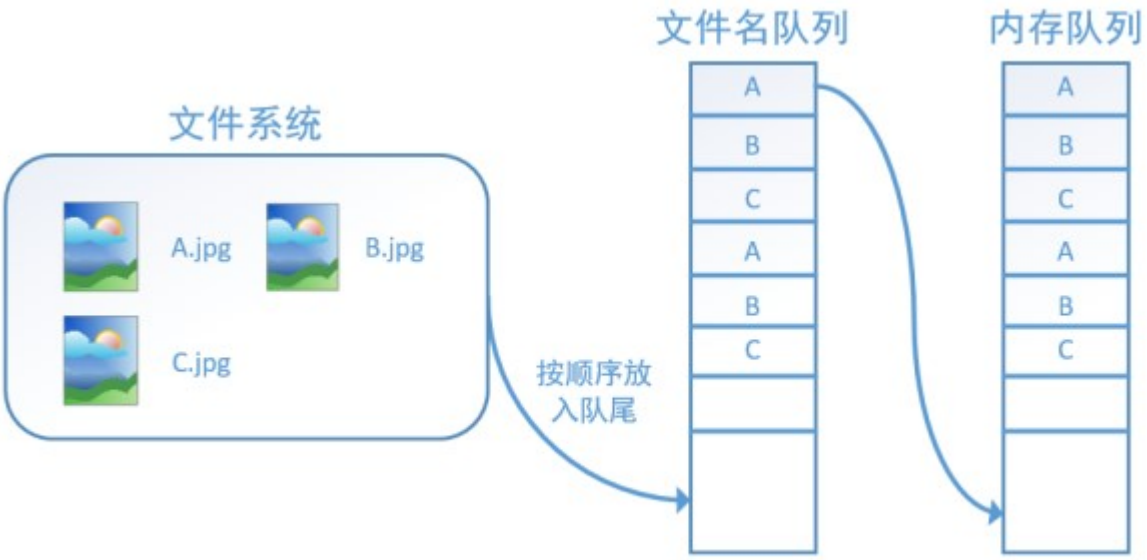
只要在文件名队列中将A、B、C依次放入两次再标记结束就可以了。

二、tensorflow读取数据机制的对应函数

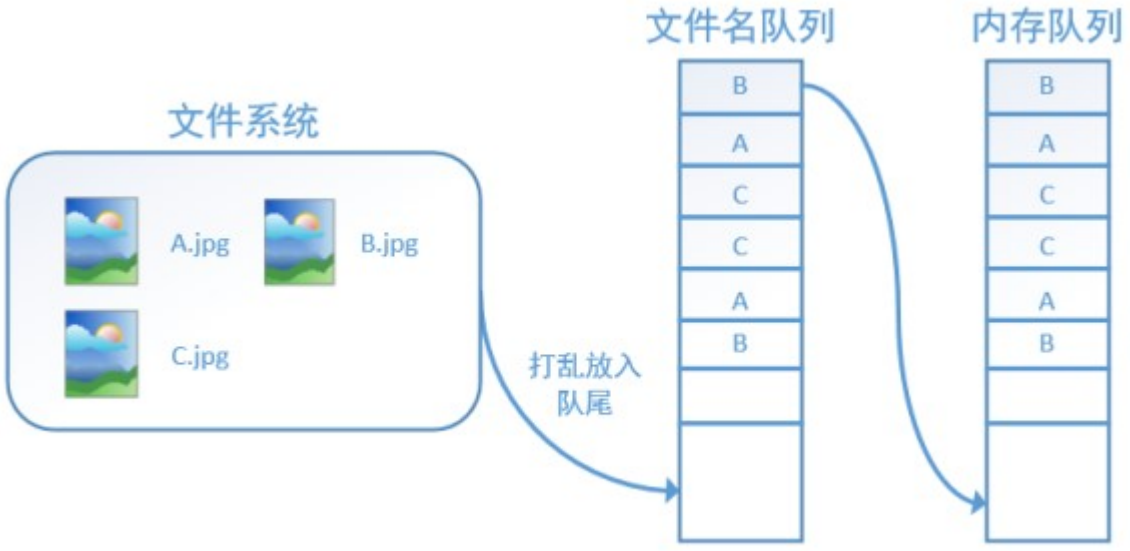
如何在tensorflow中创建上述的两个队列呢？

对于文件名队列，我们使用tf.train.string_input_producer函数。这个函数需要传入一个文件名list，系统会自动将它转为一个文件名队列。

此外tf.train.string_input_producer还有两个重要的参数，一个是num_epochs，它就是我们上文中提到的epoch数。另外一个就是shuffle，shuffle是指在一个epoch内文件的顺序是否被打乱。若设置shuffle=False，如下图，每个epoch内，数据还是按照A、B、C的顺序进入文件名队列，这个顺序不会改变：



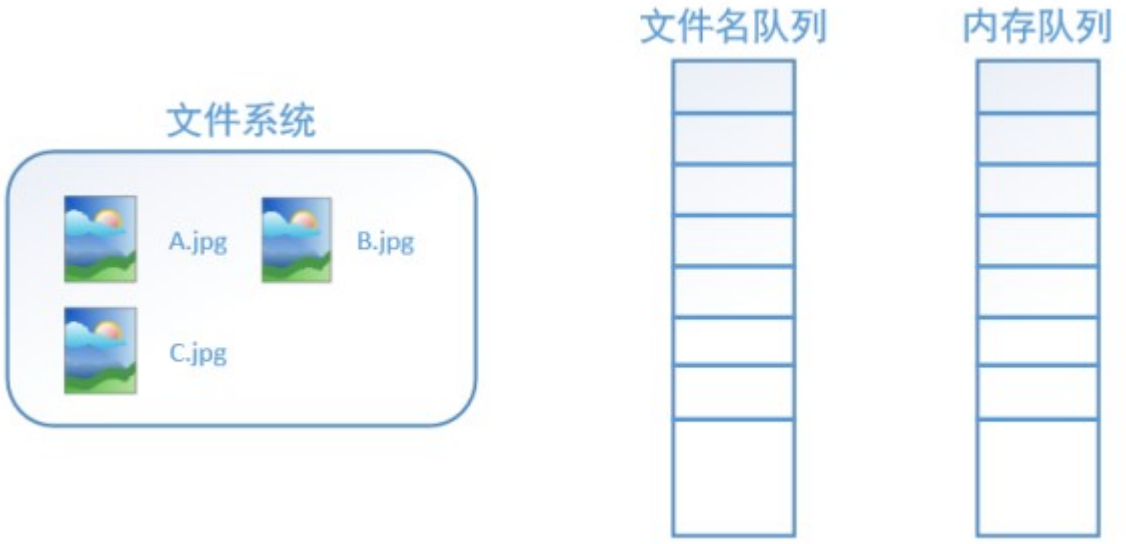
如果设置shuffle=True，那么在一个epoch内，数据的前后顺序就会被打乱，如下图所示：



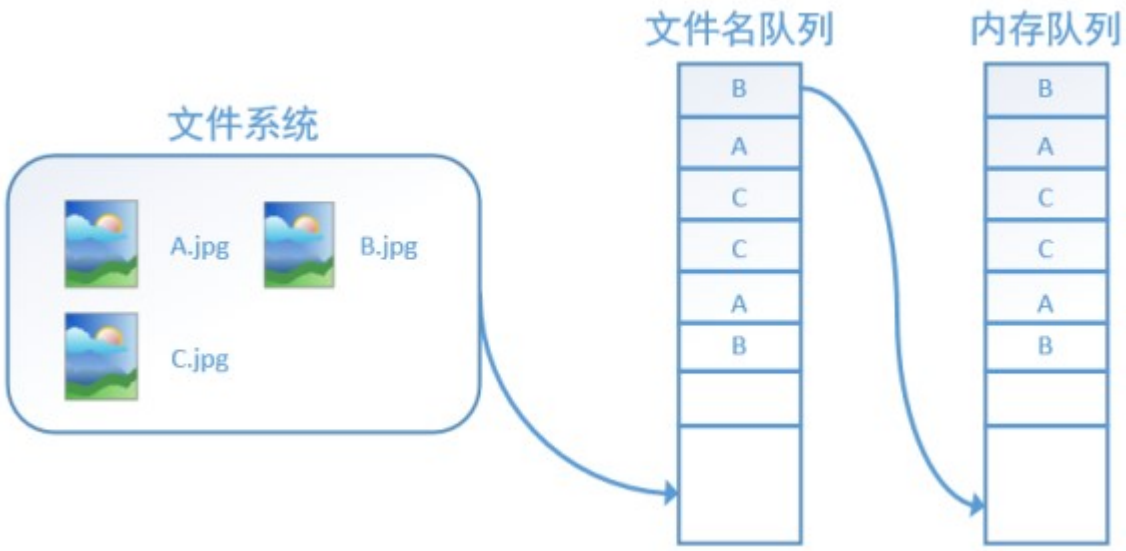
在tensorflow中，内存队列不需要我们自己建立，我们只需要使用reader对象从文件名队列中读取数据就可以了，具体实现可以参考下面的实战代码。

除了tf.train.string_input_producer外，我们还要额外介绍一个函数：tf.train.start_queue_runners。初学者会经常在代码中看到这个函数，但往往很难理解它的用处，在这里，有了上面的铺垫后，我们就可以解释这个函数的作用了。

在我们使用tf.train.string_input_producer创建文件名队列后，整个系统其实还是处于“停滞状态”的，也就是说，我们文件名并没有真正被加入到队列中（如下图所示）。此时如果我们开始计算，因为内存队列中什么也没有，计算单元就会一直等待，导致整个系统被阻塞。



而使用tf.train.start_queue_runners之后，才会启动填充队列的线程，这时系统就不再“停滞”。此后计算单元就可以拿到数据并进行计算，整个程序也就跑起来了，这就是函数tf.train.start_queue_runners的用处。



三、实战代码

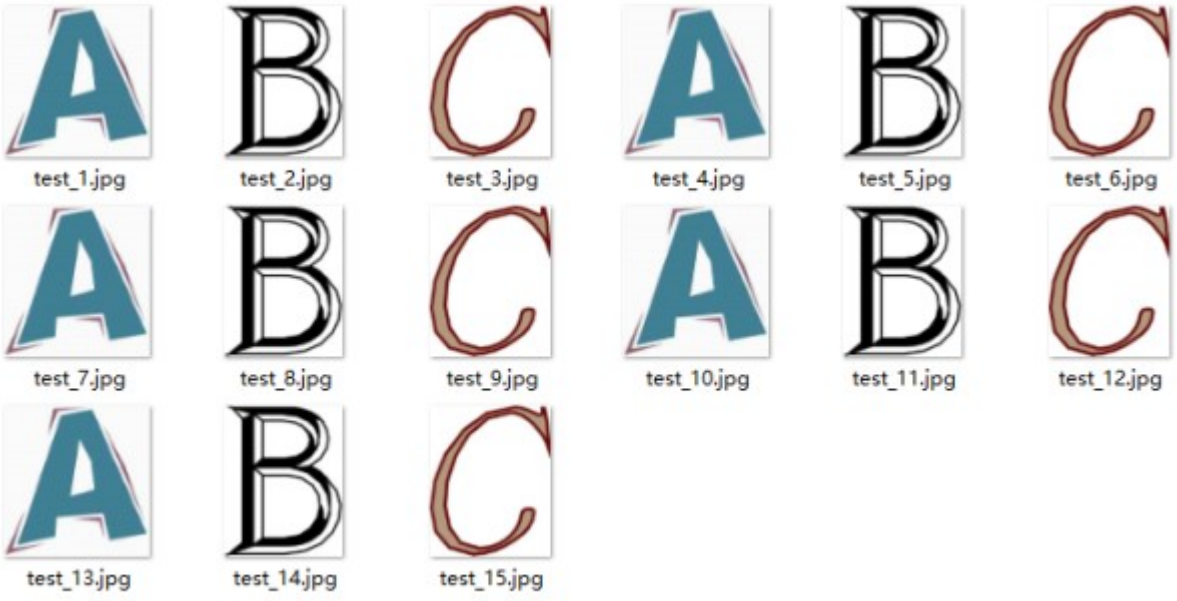
我们用一个具体的例子感受tensorflow中的数据读取。如图，假设我们在当前文件夹中已经有A.jpg、B.jpg、C.jpg三张图片，我们希望读取这三张图片5个epoch并且把读取的结果重新存到read文件夹中。

对应的代码如下：

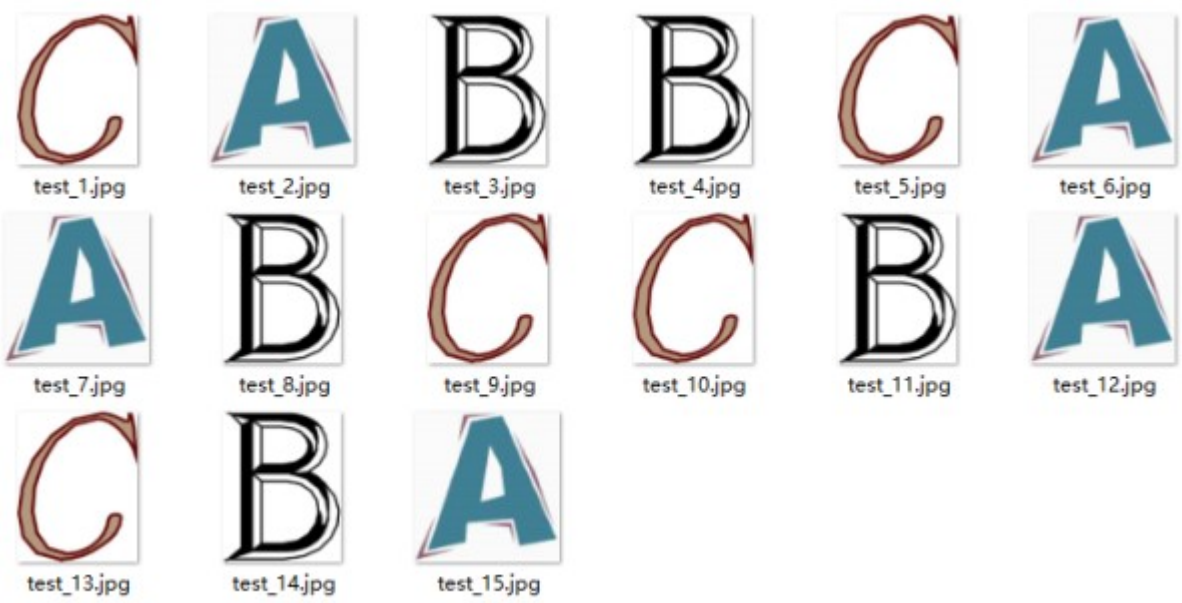
```
# 导入tensorflow
import tensorflow as tf

# 新建一个Session
with tf.Session() as sess:
    # 我们要读三幅图片A.jpg, B.jpg, C.jpg
    filename = ['A.jpg', 'B.jpg', 'C.jpg']
    # string_input_producer会产生一个文件名队列
    filename_queue = tf.train.string_input_producer(filename, shuffle=False, num_epochs=5
)
    # reader从文件名队列中读数据。对应的方法是reader.read
    reader = tf.WholeFileReader()
    key, value = reader.read(filename_queue)
    # tf.train.string_input_producer定义了一个epoch变量，要对它进行初始化
    tf.local_variables_initializer().run()
    # 使用start_queue_runners之后，才会开始填充队列
    threads = tf.train.start_queue_runners(sess=sess)
    i = 0
    while True:
        i += 1
        # 获取图片数据并保存
        image_data = sess.run(value)
        with open('read/test_%d.jpg' % i, 'wb') as f:
            f.write(image_data)
```

我们这里使用filename_queue = tf.train.string_input_producer(filename, shuffle=False, num_epochs=5)建立了一个会跑5个epoch的文件名队列。并使用reader读取，reader每次读取一张图片并保存。

运行代码后，我们得到就可以看到read文件夹中的图片，正好是按顺序的5个epoch：



如果我们设置filename_queue = tf.train.string_input_producer(filename, shuffle=False, num_epochs=5)中的shuffle=True，那么在每个epoch内图像就会被打乱，如图所示：



我们这里只是用三张图片举例，实际应用中一个数据集肯定不止3张图片，不过涉及到的原理都是共通的。